Disambiguating Grammatically Ambiguous Sentences By Asking

Masaru Tomita
Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

The problem addressed in this paper is to disambiguate grammatically ambiguous input sentences by asking the user, who need not be a computer specialist or a linguist, without showing any parse trees or phrase structure rules. Explanation List Comparison (ELC) is the technique that implements this process. It is applicable to all parsers which are based on phrase structure grammar, regardless of the parser implementation. An experimental system has been implemented at Carnegie-Mellon University, and it has been applied to English-Japanese machine translation at Kyoto University.

1. Introduction

A large number of techniques using semantic information have been developed to resolve natural language ambiguity. However, not all ambiguity problems can be solved by those techniques at the current state of art. Moreover, some sentences are *absolutely* ambiguous, that is, even a human cannot disambiguate them. Therefore, it is important for the system to be capable of asking a user questions interactively to disambiguate a sentence.

Here, we make an important condition that an user is neither a computer scientist nor a linguist. Thus, an user may not recognize any special terms or notations like a tree structure, phrase structure grammar, etc.

The first system to disambiguate sentences by asking interactively is perhaps a program called "disambiguator" in Kay's MIND system [2]. Although the disambiguation algorithm is not presented in [2], some basic ideas have been already implemented in the Kay's system[2]. In this paper, we shall only deal with grammatical ambiguity, or in other words, syntactic ambiguity. Other ambiguity problems, such as word-sense ambiguity and referential ambiguity, are excluded.

Suppose a system is given the sentence:

"Mary saw a man with a telescope"

and the system has a phrase structure grammar including the following rules <a> - <g>:

```
<a>     S --> NP + VP
<b>     S --> NP + VP + PP
<c>     NP --> *noun
<d>     NP --> *det + *noun
<e>     NP --> NP + PP
<f>     PP --> *prep + NP
<g>     VP --> *verb + NP
```

The system would produce two parse trees from the input sentence (I. using rules <b>,<c>,<g>,<d>,<f>,<d>; II. using rules <a>,<c>,<g>,<e>,<d>,<f>,<d>). The difference is whether the preposition phrase "with a telescope" qualifies the noun phrase "a man" or the sentence "Mary saw a man". This paper shall discuss on how to ask the user to select his intended interpretation without showing any kind of tree structures or phrase structure grammar rules. Our desired question for that sentence is thus something like:

1) The action "Mary saw a man" takes place "with a telescope"
2) "a man" is "with a telescope"
NUMBER ?

The technique to implement this, which is described in the following sections, is called *Explanation List Comparison*.

2. Explanation List Comparison

The basic idea is to attach an *Explanation Template* to each rule. For example, each of the rules <a> - <g> would have an explanation template as follows:

```
        Explanation Template

<a>     (1) is a subject of the action (2)
<b>     The action (1 2) takes place (3)
<c>     (1) is a noun
<d>     (1) is a determiner of (2)
<e>     (1) is (2)
<f>     (1) is a preposition of (2)
<g>     (2) is an object of the verb (1)
```

Whenever a rule is employed to parse a sentence, an explanation is generated from its explanation template. Numbers in an explanation template indicate n-th constituent of the right hand side of the rule. For instance, when the rule <f>

PP --> *prep + NP

matches "**with a telescope**" (*prep = "**WITH**"; NP = "**a**

[1]This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory Under Contract F33615-81 K-1539. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

[2]Personal communication.

telescope"). the explanation

"(with) is a preposition of (a telescope)"

is generated. Whenever the system builds a parse tree, it also builds a list of explanations which are generated from explanation templates of all rules employed. We refer to such a list as an *explanation list*. The explanation lists of the parse trees in the example above are:

Alternative I.

<b> The action (Mary saw a man) takes place (with a telescope)
<c> (Mary) is a noun
<g> (a man) is an object of the verb (saw)
<d> (A) is a determiner of (man)
<f> (with) is a preposition of (a telescope)
<d> (A) is a determiner of (telescope)

Alternative II.

<a> (Mary) is a subject of the action (saw a man with a telescope)
<c> (Mary) is a noun
<g> (a man with a telescope) is an object of the verb (saw)
<e> (a man) is (with a telescope)
<d> (A) is a determiner of (man)
<f> (with is a preposition of (a telescope)
<d> (A) is a determiner of (telescope)

In order to disambiguate a sentence, the system only examines these Explanation Lists, but not parse trees themselves. This makes our method independent from internal representation of a parse tree. Loosely speaking, when a system produces more than one parse tree, explanation lists of the trees are "compared" and the "difference" is shown to the user. The user is, then, asked to select the correct alternative.

3. The revised version of ELC

Unfortunately, the basic idea described in the preceding section does not work quite well. For instance, the difference of the two explanation lists in our example is

1)
The action (Mary saw a man) takes place (with a telescope),
(a man) is an object of the verb (saw);
2)
(Mary) is a subject of the action (saw a man with a telescope),
(a man with a telescope) is an object of the verb (saw),
(a man) is (with a telescope);

despite the fact that the essential difference is only

1) The action (Mary saw a man) takes place (with a telescope)
2) (a man) is (with a telescope)

Two refinement ideas, *head* and *multiple explanations*, are introduced to solve this problem.

3.1. Head

We define *head* as a word or a minimal cluster of words which are syntactically dominant in a group and could have the same syntactic function as the whole group if they stood alone. For example, the head of "VERY SMART PLAYERS IN NEW YORK" is "PLAYERS", and the head of "INCREDIBLY BEAUTIFUL" is "BEAUTIFUL", but the head of "I LOVE CATS" is "I LOVE CATS" itself. The idea is that, whenever the system shows a part of an input sentence to the user, only the head of it is shown. To implement this idea, each rule must have a *head definition* besides an explanation template, as follows.

Rule	Head
<a>	[1 2]
<b>	[1 2]
<c>	[1]
<d>	[1 2]
<e>	[1]
<f>	[1 2]
<g>	[1 2]

For instance, the head definition of the rule <b> says that the head of the construction "NP + VP + PP" is a concatenation of the head of 1-st constituent (NP) and the head of 2-nd constituent (VP). The head of "A GIRL with A RED BAG saw A GREEN TREE WITH a telescope" is, therefore, "A GIRL saw A TREE", because the head of "A GIRL with A RED BAG" (NP) is "A GIRL" and the head of "saw A GREEN TREE" (VP) is "saw A TREE".

In our example, the explanation

(Mary) is a subject of the action (saw a man with a telescope)

becomes

(Mary) is a subject of the action (saw a man),

and the explanation

(a man with a telescope) is an object of the verb (saw)

becomes

(a man) is an object of the verb (saw),

because the head of "saw a man with a telescope" is "saw a man", and the head of "a man with a telescope" is "a man".

The difference of the two alternatives are now:

1)
The action (Mary saw a man) take place (with a telescope);
2)
(Mary) is a subject of the action (saw a man),
(a man) is (with a telescope);

3.2. Multiple explanations

In the example system we have discussed above, each rule generates exactly one explanation. In general, multiple explanations (including zero) can be generated by each rule. For example, rule <b>

```
S --> NP + VP + PP
```

should have two explanation templates:

```
(1) is a subject of the action (2)
The action (1 2) takes place (3),
```

whereas rule <a>

```
S --> NP + VP
```

should have only one explanation template:

```
(1) is a subject of the action (2).
```

With the idea of head and multiple explanations, the system now produces the ideal question, as we shall see below.

3.3. Revised ELC

To summarize, the system has a phrase structure grammar, and each rule is followed by a head definition followed by an arbitrary number of explanation templates.

Rule	Head	Explanation Template
<a>	[1 2]	(1) is a subject of the action (2)
<b>	[1 2]	(1) is a subject of the action (2) The action (1 2) takes place (3)
<c>	[1]	<<none>>
<d>	[1 2]	(1) is a determiner of (2)
<e>	[1]	(1) is (2)
<f>	[1 2]	(1) is a preposition of (2)
<g>	[1 2]	(2) is an object of the verb (1)

With the ideas of head and multiple explanation, the system builds the following two explanation lists from the sentence "Mary saw a man with a telescope".

Alternative I.

<b> (Mary) is a subject of the action (saw a man)
<b> The action (Mary saw a man) takes place (with a telescope)
<g> (a man) is an object of the verb (saw)
<d> (A) is a determiner of (man)
<f> (with) is a preposition of (a telescope)
<d> (A) is a determiner of (telescope)

Alternative II.

<a> (Mary) is a subject of the action (saw a man)
<g> (a man) is an object of the verb (saw)
<e> (a man) is (with a telescope)
<d> (A) is a determiner of (man)
<f> (with is a preposition of (a telescope)
<d> (A) is a determiner of (telescope)

The difference between these two is

The action (Mary saw a man) takes place (with a telescope)

and

(a man) is (with a telescope).

Thus, the system can ask the ideal question:

1) The action (Mary saw a man) takes place (with a telescope)
2) (a man) is (with a telescope)
Number?

4. More Complex Example

The example in the preceding sections is somewhat oversimplified, in the sense that there are only two alternatives and only two explanation lists are compared. If there were three or more alternatives, comparing explanation lists would be not as easy as comparing just two.

Consider the following example sentence:

Mary saw a man in the park with a telescope.

This sentence is ambiguous in 5 ways, and its 5 explanation lists are shown below.

Alternative I.

(a man) is (in the park)
(the park) is (with a telescope)
: :
: :

Alternative II.

(a man) is (with a telescope)
(a man) is (in the park)
: :
: :

Alternative III.

The action (Mary saw a man) takes place (with a telescope)
(a man) is (in the park)
: :
: :

Alternative IV.

The action (Mary saw a man) takes place (in the park)
(the park) is (with a telescope)
: :
: :

Alternative V.

The action (Mary saw a man) takes place (with a telescope)
The action (Mary saw a man) takes place (in the park)
: :
: :

With these 5 explanation lists, the system asks the user a question twice, as follows:

1) (a man) is (in the park)
2) The action (Mary saw a man) takes place (in the park)
NUMBER? 1

1) (the park) is (with a telescope)
2) (a man) is (with a telescope)
3) The action (Mary saw a man) takes place (with a telescope)
NUMBER? 3

The implementation of this is described in the following.

We refer to the set of explanation lists to be compared, $\{L_1, L_2, \ldots\}$, as A. If the number of explanation lists in A is one ; just return the parsed tree which is associated with that explanation list. If there are more than one explanation list in A, the system makes a *Qlist* (Question list). The Qlist is a list of explanations

$$\text{Qlist} = \{ e_1, e_2, \ldots, e_n \}$$

which is shown to the user to ask a question as follows:

```
1)  e_1
2)  e_2
 .    .
 .    .
n)  e_n
Number?
```

Qlist must satisfy the following two conditions to make sure that always exactly one explanation is true.

- Each explanation list L in A must contain at least one explanation e which is also in Qlist. Mathematically, the following predicate must be satisfied.

  $$\forall L \exists e (e \in L \wedge e \in \text{Qlist})$$

  This condition makes sure that at least one of explanations in a Qlist is true.

- No explanation list L in A contains more than one explanation in a Qlist. That is,

$$\neg(\exists L \exists e \exists e'(L \in A \wedge e \in L \wedge e' \in L \wedge e \in \text{Qlist} \wedge e' \in \text{Qlist} \wedge e \neq e')$$

This condition makes sure that at most one of explanations in Qlist is true.

The detailed algorithm of how to construct a Qlist is presented in Appendix.

Once a Qlist is created, it is presented to the user. The user is asked to select one correct explanation in the Qlist, called the *key explanation*. All explanation lists which do not contain the key explanation are removed from A. If A still contains more than one explanation list, another Qlist for this new A is created, and shown to the user. This process is repeated until A contains only one explanation list.

5. Concluding Remarks

An experimental system has been written in Maclisp, and running on Tops-20 at Computer Science Department, Carnegie-Mellon University. The system parses input sentences provided by a user according to grammar rules and a dictionary provided by a super user. The system, then, asks the user questions, if necessary, to disambiguate the sentence using the technique of Explanation List Comparison. The system finally produces only one parse tree of the sentence, which is the intended interpretation of the user. The parser is implemented in a bottom-up, breath-first manner, but the idea described in the paper is independent from the parser implementation and from any specific grammar or dictionary.

The kind of ambiguity we have discussed is *structural* ambiguity. An ambiguity is structural when two different structures can be built up out of smaller constituents of the same given structure and type. On the other hand, an ambiguity is *lexical* when one word can serve as various parts of speech. Resolving lexical ambiguity is somewhat easier, and indeed, it is implemented in the system. As we can see in the Sample Runs below, the system first resolves lexical ambiguity in the obvious manner, if necessary.

Recently, we have integrated our system into an English-Japanese Machine Translation system [3], as a first step toward user-friendly interactive machine translation [6]. The interactive English Japanese machine translation system has been implemented at Kyoto University in Japan [4, 5].

Acknowledgements

I would like to thank Jaime Carbonell, Herb Simon, Martin Kay, Jun-ich Tsujii, Toyoaki Nishida, Shuji Doshita and Makoto Nagao for thoughtful comments on an earlier version of this paper.

Appendix A: Qlist-Construction Algorithm

```
input $A$ : set of explanation lists
output Qlist : set of explanations
local $e$ : explanation
      $L$ : explanation list (set of explanations)
      $U, C$ : set of explanation lists

1: $C \Leftarrow \phi$
2: $U \Leftarrow A$
3: Qlist $\Leftarrow \phi$
4: if $U = \phi$ then return Qlist
5: select one explanation $e$ such that
     $e$ is in some explanation list $\in U$,
     but not in any explanation list $\in C$;
     if no such $e$ exists, return ERROR
6: Qlist $\Leftarrow$ Qlist + $\{e\}$
7: $C \Leftarrow C + \{L \mid e \in L \wedge L \in U\}$
8: $U \Leftarrow \{L \mid e \notin L \wedge L \in (U)\}$
9: goto 4
```

- The input to this procedure is a set of explanation lists, $\{L_1, L_2, \ldots\}$.
- The output of this procedure is a list of explanations, $\{e_1, e_2, \ldots, e_n\}$, such that each explanation list, L_i, contains exactly one explanation which is in the Qlist.
- An explanation list L is called *covered*, if some explanation e in L is also in Qlist. L is called *uncovered*, if any of the explanations in L is not in Qlist. C is a set of covered explanation lists in A, and U is a set of uncovered explanation lists in A.
- 1-3: initialization. Let Qlist be empty. All explanation lists in A are uncovered.
- 4: if all explanation lists are covered, quit.
- 5-6: select an explanation e and put it into Qlist to cover some of uncovered not explanation lists. e must be such that it does exist in any of covered explanation lists (if it does exist, the explanation list has two explanation in A, violating the Qlist condition).
- 7-8: make uncovered explanation lists which are now covered by e to be covered.
- 9: repeat the process until everything is covered.

References

[1] Kay, M.
The MIND System.
Algorithmic Press, New York, 1973, .

[2] Nishida, T. and Doshita, S.
An Application of Montague Grammar to English-Japanese Machine Translation.
Proceedings of conference on Applied Natural Language Processing :156-165, 1983.

[3] Tomita, M., Nishida, T. and Doshita, S.
An Interactive English-Japanese Machine Translation System.
Forthcoming (in Japanese), 1984.

[4] Tomita, M., Nishida, T. and Doshita, S.
User Front-End for disambiguation in Interactive Machine Translation System.
In *Tech. Reports of WGNLP.* Information Processing Society of Japan, (in Japanese, forthcoming), 1984.

[5] Tomita, M.
The Design Philosophy of Personal Machine Translation System.
Technical Report, Computer Science Department, Carnegie-Mellon University, 1983.

Appendix B: Sample Runs

```
(transline '(time flies like an arrow in Japan))

(---END OF PARSE-- 10 ALTERNATIVES)

(The word  TIME (1) is:)
(1 : VERB)
(2 : NOUN)
NUMBER> 2

(The word  FLIES (2) is:)
(1 : VERB)
(2 : NOUN)
NUMBER> 1

(1 : (AN ARROW) IS (IN JAPAN))
(2 : THE ACTION (TIME FLIES) TAKES PLACE (IN JAPAN))
NUMBER> 2

(S (NP (TIME *NOUN))
           (FLIES *VERB)
           (PP (LIKE *PREPOSITION) (NP (AN *DETERMINER) (ARROW *NOUN)))
           (PP (IN *PREPOSITION) (JAPAN *NOUN)))

(transline '(Mary saw a man in the apartment with a telescope))

(---END OF PARSE-- 5 ALTERNATIVES)

(1 : (A MAN) IS (IN THE APARTMENT))
(2 : THE ACTION (MARY SAW A MAN) TAKES PLACE (IN THE APARTMENT))
NUMBER> 1

(1 : (A MAN) IS (WITH A TELESCOPE))
(2 : (THE APARTMENT) IS (WITH A TELESCOPE))
(3 : THE ACTION (MARY SAW A MAN) TAKES PLACE (WITH A TELESCOPE))
NUMBER> 3

(S (NP (MARY *NOUN))
           (VP (SAW *VERB)
               (NP (NP (A *DETERMINER) (MAN *NOUN))
                   (PP (IN *PREPOSITION)
                       (NP (THE *DETERMINER) (APARTMENT *NOUN))))
           (PP (WITH *PREPOSITION)
               (NP (A *DETERMINER) (TELESCOPE *NOUN))))
```